Attack Path Generation Based on Attack and Penetration Testing Knowledge

Florian Sommer
Institute of Energy Efficient Mobility
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
e-mail: florian.sommer@h-ka.de

Reiner Kriesten
Institute of Energy Efficient Mobility
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
e-mail: reiner.kriesten@h-ka.de

***Abstract*—To protect modern vehicles against security attacks, new standards, such as ISO/SAE 21434, and regulations, such as UN R155, require security testing activities during development. For this purpose, penetration testing is often used, which is a manually performed, experience-based, and explorative test method. Due to the high complexity of modern vehicles, manual penetration testing methods reach their limits. As a result, potential vulnerabilities could be overlooked and thus remain in the vehicle. In case of a security attack, this can endanger passengers and road traffic participants. So far, penetration testing has been considered as difficult to automate, since it is an experience-based method. This paper presents a model-based approach which aims close that gap. Our approach uses knowledge of existing security attacks on vehicles to automate the security testing process. We apply our attack database (361 attacks, consisting of 621 attack steps) to a formal security model to automatically derive attack paths for testing. We also present a proposal of how this method can be transferred to derive attack paths based on knowledge and experience of penetration testers.**

***Keywords*—security testing; automation; tester experience.**

I. Introduction

The increasing complexity of modern vehicles and the growing number of automotive security attacks [1] in recent years have led to automotive security becoming a high priority in industry and research. As a result of this trend, the ISO/SAE 21434 [2] and UN R155 [3] were published. Both documents require vehicle manufacturers to comply with a specific security process during the development and life cycle of vehicles. In addition to performing a Threat Analysis and Risk Assessment (TARA) [2], as well as the derivation of security requirements and measures, a focus is also on the verification and validation of security. The latter usually takes place within a security test process. For this purpose, ISO/SAE 21434 proposes in particular an execution of penetration tests.

Problem: Penetration testing is an experience-based and explorative method, which is carried out late in development. Thus, the vehicle and its components have already been developed at that point. As a result, potential vulnerabilities can only be identified at a late stage. This type of testing is often carried out by third parties as manual black-box or grey-box tests. Modern vehicles are highly complex systems and there is usually a limited time frame available for testing. Thus, Marksteiner et al. [4] see a risk that manual penetration testing reaches its limits regarding comprehensive testing. This implies that vulnerabilities could be overlooked or not captured by testing and thus remain in the vehicle.

Solution: To face these challenges, we propose a model-based approach using knowledge and experience from past security attacks and penetration tests of vehicles. For this purpose, attack paths are automatically generated and used in security testing. This allows an early execution of testing activities in vehicle development. Our approach can be used in the context of penetration testing to systematically support testers by providing attack paths based on successful real-world attacks. This allows the security test process to be partially automated by using knowledge and experience of attacks and penetration tests. Our method can further be used to estimate the effort of test activities.

Contribution: In this publication, we present a model-based security testing method. For this purpose, a security model of a vehicle E/E architecture is created based on our past work [5] [6]. Our model can be examined for possible attack paths based on real-world attacks by applying our automotive attack database [7], which currently includes 361 attacks (consisting of 621 attack steps). Further, this approach enables us to find new attack paths by permutation of existing attack steps from the database. We also present a proposal of how this approach can be used to capture and reuse experience of penetration testers to achieve partial automation of the security test process.

This paper is structured as follows: In Section II, we describe fundamentals of security testing and model-based security testing. In Section III, the approach of this work and the creation of a security model is presented. Section IV shows how our attack database can be used to automatically derive attack paths from the security model. Section V presents a proposal how the experience of penetration testers can be captured and reused for automatic attack path generation. A discussion about the feasibility of this method and challenges is given in Section VI. In Section VII, we draw a conclusion and give an outlook on future work.

II. Background and Related Work

In this section, we provide a brief overview of security testing and model-based security testing and their application in the automotive context.

A. Security Testing

Security testing examines a system for security weaknesses [8]. This is generally done in two ways. The first way concerns functional or positive security testing [9]. This

 ISBN: 978-1-61208-996-6

typically involves testing functional security mechanisms (for example, encryption and authentication of messages) for correct functionality. This can be done as part of the traditional testing process based on requirements of security mechanisms. The second way concerns non-functional and negative security testing [9]. This type of testing is also called security vulnerability testing and is often performed through penetration tests. The tester takes the role of an attacker and attempts to find vulnerabilities in a system by carrying out security attacks [8]. Penetration testing represents an experience-based and exploratory testing method. Tests are usually performed as black-box (without system knowledge) or grey-box (partial system knowledge) tests. Several penetration testing standards exist to support testers in a structured way. Examples are Penetration Testing Execution Standard (PTES) [10] and Open Web Application Security Project (OWASP) [11] Testing Guide. For the automotive sector, the Automotive Security Testing Methodology (ASTM) [12] can be applied.

B. Security Testing in the Automotive Domain

In 2021, ISO/SAE 21434 [2] and UN R155 [3] were published for the automotive sector. These documents demand that security should be addressed throughout the development and life cycle of a vehicle. With regard to security testing, ISO/SAE 21434 in particular proposes an execution of functional testing, vulnerability scanning, fuzz testing, and penetration testing. The application of these testing techniques to automotive systems has been a subject of several recent publications. Bayer et al. [13] analyze the mentioned test methods and show potential use cases based on specific automotive technologies, such as Controller Area Network (CAN) [14], or protocols, such as Unified Diagnostic Services (UDS) [15]. Smith [16] provides a complete guide on penetration testing in vehicles. Various attack techniques on bus and diagnostic protocols, wireless communication systems, Electronic Control Units (ECUs), etc. are explained in detail. Further standards and publications propose the consideration of threat modeling for penetration testing. In this context, Dürrwang et al. [17] were able to uncover a critical vulnerability in an airbag ECU, which could lead to an unauthorized airbag deployment.

C. Model-Based Security Testing in the Automotive Domain

Model-based testing enables early testing and automation of the test process [18]. For this purpose, a System Under Test (SUT) is defined, which is commonly represented as a formal model (e.g., as a state machine). By applying test selection criteria, such as coverage criteria, test cases are derived and executed on the system. With appropriate tooling, many parts of this process can be automated. Model-based security testing combines this process with traditional security testing. The models of the SUT are extended by security-specific aspects, such as security properties, risk values, vulnerabilities, or security mechanisms. Resulting models are used to derive security test cases or attack paths. An example of a model-based security test method in the automotive domain is presented by Cheah et al. [19]. Here, attack trees, which emerge as part of a threat modeling process, are formally described by Communicating Sequential Processes (CSP) [20]. From the resulting model, a refinement checking tool is used to derive test cases to test a Bluetooth [21] device. Further, Oruganti et al. [22] and Appel et al. [23] present approaches based on Matlab/Simulink models for hardware-in-the-loop testing. Volkersdorfer et al. [24] present a model-based security approach using attack and adversary models to simulate attacks on a specific attack target. This approach is demonstrated using two application scenarios: attacks on a user's access data to a web application and the manipulation of an automotive ECU.

The presented related work focuses on finding attacks or test cases which reveal vulnerabilities in a system. This is done in an exploratory, but also in a systematic, guided, and model-based way. The authors use information about the system and its functionality and analyze how these systems can be attacked/tested. In comparison, our approach is based on a formal vehicle network model, which is specified in a generic way. This allows an application of a wide range of different attacks. For this purpose, we combine collected knowledge of attackers or security testers within a database containing successful real-world attacks. These serve as a basis for analyzing and generating attack paths as part of the security testing process.

III. Approach and Modeling Process

Penetration testing is an experience-based testing method which leverages an attacker's perspective to compromise and test systems. Therefore, knowledge about security attacks and how they are executed is an important source of information for a tester. In this section, a model-based approach is presented using knowledge and experience of attackers and penetration testers to automatically generate attack paths for security testing. Our approach and its overall process is illustrated in Figure 1. First, a security model is generated based on an Electrical/Electronic (E/E) architecture. Since we cover security testing in our approach, we need to consider all entities of the E/E architecture which have an impact on the cyber security of a vehicle. This especially involves ECUs, sensors, actuators, software applications, communication systems, and interfaces. These elements and their interactions are enhanced with security-specific aspects to create a security model. By applying our attack database [7], the model can be analyzed for possible attack paths based on successful real-world attacks. Furthermore, attack paths can be derived and adapted to the vehicle under test. In the following sections, we explain that process and further introduce details on how this approach can be implemented based on experience of penetration testers. In the first step, we build a model which represents both the network architecture of a vehicle and security-specific properties. Therefore, we build on our previous work [5], in which we introduced our concept of *Attacker Privileges*. These privileges represent abstract states an attacker can achieve in a system by exploiting vulnerabilities. Thus, we are able to introduce attack paths in our model.

 ISBN: 978-1-61208-996-6

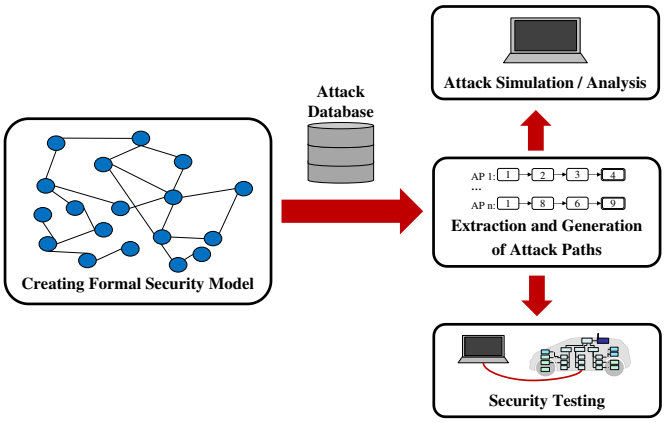

Figure 1. Model-based approach for security testing. A security model is created which is used to derive attack paths for the SUT. These paths can be used for attack simulation/analysis and security testing.

We distinguish five privileges as illustrated in Figure 2. The *Read/Write* privilege describes an attackers ability to read and write data or messages on a communication channel. By acquiring the *Execute* privilege an attacker is able to trigger implemented functions on a component (e.g., controlling an actuator via diagnostic functions). The *Read* privilege enables an attacker to extract data or information from a component (e.g., extracting secret keys). The *Write* privilege describes an ability to write or change data on a component (e.g., deleting error logs). By acquiring the *Full Control* privilege an attacker has total control over a component (e.g., by updating an ECU with malicious software). All five privileges can be assigned to elements of a vehicle network as shown in Figure 2. The *Read/Write* privilege can only be assigned to communication systems (e.g., CAN or Bluetooth). The other four privileges are assigned to components (e.g., ECUs or sensors). If an attacker reaches the *Read/Write* privilege (PL1, Figure 2 left), any other privilege (PL2 - PL5) on a component can be acquired from there if corresponding vulnerabilities are exploited. On the component, the attacker is able to switch between privileges PL2 - PL5 when exploiting vulnerabilities. We also assume that it is possible for an attacker to access communication interfaces of a component once PL4 or PL5 has been reached. This would allow access to further connected communication systems (PL1, Figure 2 right). Applying the *Attacker Privileges* to an entire E/E architecture of a vehicle thus allows to compose chains of privileges an attacker can reach. Since the successful exploitation of a vulnerability is necessary to achieve a privilege, attack paths within a vehicle network can be modeled. In [5], we used this approach to create a formal transition system based on a vehicle network. This model was used to automatically generate attack trees by applying model checking techniques in the context of threat modeling. The resulting attack tree contained a critical real-world airbag vulnerability. Thus, we could show that our *Attacker Privileges* are able to represent critical attack paths in vehicle networks. In this paper, we apply this approach in the context of a model-based security test method we introduced in [6]. For this purpose, we assume a simple E/E architecture example.

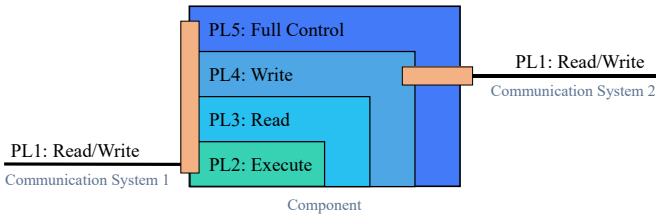

Figure 2. Distribution of *Attacker Privileges* to elements of a vehicular network [5].

In this example, an On-Board Diagnostics (OBD) interface is connected to a Central Gateway (CGW) via a CAN bus (CAN 1). The CGW is also connected to another CAN bus (CAN 2). Applying the *Attacker Privileges* to that network results in a security model shown in Figure 3. The illustrated security model corresponds to the graphical representation of an Extended Finite State Machine (EFSM) [25] we use for formalization. For the two CAN buses and the OBD interface, one state was defined to each which assigns the *Read/Write* privilege. For the CGW, four states were defined which correspond to the remaining four privileges. Security mechanisms are not considered here for reasons of simplicity. To create transitions, it was assumed that an attacker or tester has access to the OBD interface and wants to gain access to the internal vehicle network via the CGW. To model transitions between the states, we apply our privilege model as illustrated in Figure 2. This results in a transition from *State 1* to *State 2* and from *State 2* to *State 3* - *State 6* respectively. It can be switched arbitrarily between the states of the CGW using a corresponding transition. The only exception here is *State 6*, since we assume that the *Full Control* privilege includes the other three privileges. Finally, a transition leads from *State 5* and *State 6* to *State 7* as explained in Figure 2. A formal description of the EFSM presented in Figure 3 is not further discussed here. Only the syntax and semantics of transitions is revisited in the next section to explain our concept for attack path generation based on attacker behavior and penetration tester experience.

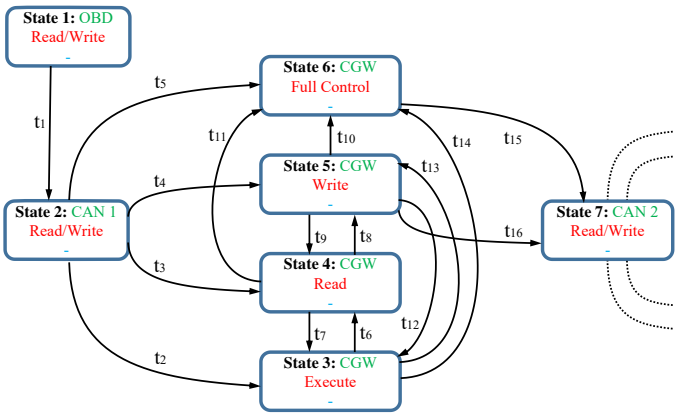

Figure 3. Security model based on an E/E architecture consisting of an OBD interface, a Central Gateway (CGW), and two CAN buses.

 ISBN: 978-1-61208-996-6

IV. Attack Path Generation using an Attack Database

In this section, we explain how attacker behavior based on our attack database [1] can be used to derive attack paths from the security model presented in Section III. First, we explain the transitions within our model. In general, transitions within an EFSM have the following structure:

$$Source \xrightarrow{Event\,[Guard]\,/\,Action} Target$$

A transition enables the change from a source state to a target state. The transition is triggered when an *Event* occurs (e.g., reception of a message or an input). In addition, a *Guard* condition must be met (e.g., message has a matching identifier) for a state change to occur. If an event occurs and the guard condition is met, the state will change and an output *Action* (e.g., sending an acknowledgement) will be triggered. We use these semantics to model an occurrence of attacks (exploitation of a vulnerability) in our security model. This results in the following structure as an example application to the transition t_2 from Figure 3:

$$State\,2 \xrightarrow{\substack{Exploit\,[Vulnerability]\,/\\ \{Privilege,\,Violated\,Security\,Property\}}} State\,3$$

To get from *State 2* (CAN 1 with *Read/Write* privilege) to *State 3* (CGW with *Execute* privilege), an attacker must employ an *Exploit* based on a *Vulnerability* leading to this state. The question now is, which exploits and which vulnerabilities can be used and how are they described. This is where our attack database [7] is applied, which currently (as of June 2022) contains 361 publicly known security attacks on vehicles. Overall, these attacks consist of 621 individual attack steps. To provide a uniform description of these attacks and steps, we published an attack taxonomy in [1] and classified our database accordingly. This taxonomy has different categories for describing an attack step, such as used tools, interfaces, brief description of the attack, requirements and restrictions, etc. For the transitions of the security model presented in this paper, the taxonomy categories shown in Figure 4 are particularly relevant. For each database attack step, there is a category *Component* and *Interface*, which specify affected components (e.g., ECU, Sensor, or Actuator) or interfaces (e.g., OBD or CAN interface). Furthermore, the *Violated Security Property* and the achieved *Attacker Privilege* are given for each step. To describe the *Vulnerability*, we use the Common Weakness Enumeration (CWE™) [26] provided by *The MITRE Corporation*. CWE is a systematic and hierarchically classified listing of software and hardware weaknesses, which are also used, for example, in the National Vulnerability Database (NVD) [27]. To describe exploits of a transition, we use the STRIDE classification [28]. STRIDE divides an attack into the categories *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege*.

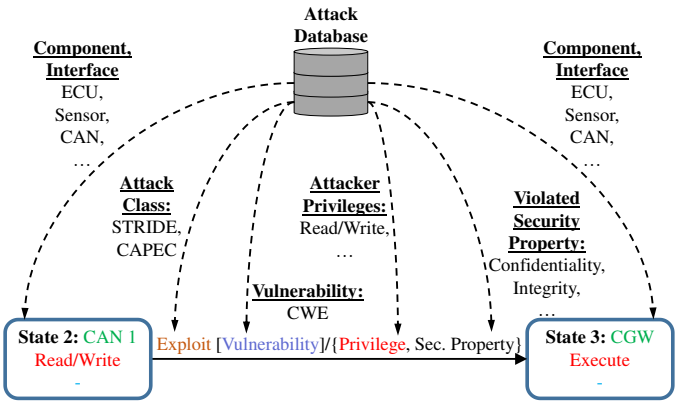

Figure 4. General application of the attack database to transitions of the security model.

In addition, we use the Common Attack Pattern Enumeration and Classification (CAPEC™) [29], which like the CWE is provided by MITRE. CAPEC provides a hierarchical description scheme for attack patterns (attack techniques/methods). These have a direct link to the CWE elements, which could be exploited by a respective attack pattern. The application of CAPEC is diverse. Currently, CAPEC suggests 26 different use cases. One of which is using attack patterns as a metric to comply with standards. Thus, CAPEC can be used to comply with automotive threats of the UN R155. All elements of a transition can be described by data of our attack database. This allows an application of all database attack steps to the security model. Thus, the model can be analyzed for the presence of real-world attack paths. In principle, this can be done in two ways. On the one hand, it can be checked whether an attack path is found exactly as described in the database. For example, if an attack consists of four attack steps, it can be analyzed whether these four explicit steps and their order can be mapped to the model. On the other hand, it is also possible to search for attack paths in the model which are composed of attack steps of several different attacks. This makes it possible to find new attack paths in our model based on the permutation of existing attack steps.

Since our security model is based on a formal EFSM, the entire process from model creation to analysis and generation of attack paths can be completely automated through a software tool. For this purpose, the E/E architecture of a vehicle, implemented security mechanisms, and the *Attacker Privileges* have to be provided. We plan on creating such a tool in future work. This would allow an attack or vulnerability analysis to take place at model level in an automated way (for example, by employing search algorithms). Furthermore, concrete attack paths to the vehicle under test can be derived, which can be used for security or penetration testing.

V. Attack Path Generation based on Penetration Tester Experience

In the previous section, we demonstrated how existing attacks from our attack database can be used to analyze a security model for existing attack paths. In this section, we

 ISBN: 978-1-61208-996-6

present a proposal on how that process can be used to derive attack paths based on experience of penetration testers. Since penetration testing is experience-based and explorative, it is hard to be automated. The main problem is how to handle and capture the experience of a tester. A tester usually gains experience by performing several penetration tests on different systems. This builds up knowledge about which systems are more likely to have vulnerabilities, or which attack techniques are more likely to succeed. For example, the ECUs of a particular vendor might be more vulnerable against buffer overflow attacks. Thus, in this case the tester would first try to execute buffer overflow attacks to other ECUs of that vendor in further tests of other vehicles. This accumulated knowledge is used again in subsequent tests, i.e., a tester first tries to find known vulnerabilities in the system based on his experience. In order to capture that experience, the attack database from the previous section should first be examined again. This database is a collection of successful security attacks on vehicles. These were almost exclusively carried out from an attacker's perspective with little knowledge of the vehicle systems, even in cases where attacks were carried out by security researchers. The attack database can thus be seen as a collection of attacker experience, behavior, and knowledge. If there is a database containing successful penetration tests instead of or in addition to the attack database, the experience and knowledge of penetration testers can be captured in the same way. Such a database could be maintained by testers, for example, within a penetration testing vendor, in order to use it in the same way as the attack database (see Section IV). Creating that database can be done iteratively over several penetration tests. Our idea is inspired by the capture-replay principle from testing Graphical User Interface (GUI) applications as described by Liu et al. [30]. Here, inputs made manually by a user in a GUI application are logged and transferred to test scripts. These can then be reused for new and automated test executions. In Figure 5, this process is illustrated for our automotive use case. The security model (see Section IV) of the vehicle under test could be made available to penetration testers within a software tool. The tester uses his experience to exploratively find vulnerabilities in the SUT through appropriate testing, attacks, etc.

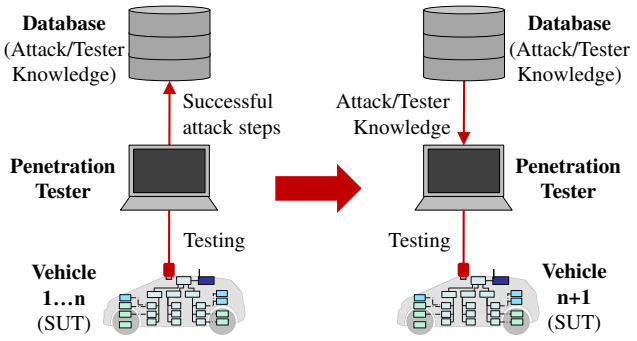

Figure 5. Collecting successful penetration testing attack steps in order to reuse that knowledge in new penetration tests.

Successful attack steps are then logged/recorded by the tester in the model, i.e., exploited paths are selected in the model and respective information (e.g., a specific attack technique used and vulnerability exploited) is specified for each attack step. Successful attack paths are then transferred to a database. If this process is carried out over several tests or different testers, an experience-based penetration testing knowledge database can be created. This can be used within model-based testing methodologies (as in the previous section) and associated tools, or within an expert system to support penetration testers in future testing. We also see a benefit of this approach for novices just entering the security testing domain, as they can benefit from an accumulated knowledge of experienced testers.

VI. Discussion

In this section, the presented approach is discussed to address use cases, current challenges, and limitations. In order to derive attack paths from a security model, only a vehicle's E/E architecture and an attack/tester database are required. For this reason, our method can be used at an early stage in the development process. This enables an application even previous to penetration testing, e.g., at the integration and system test stage. In this way, potential vulnerabilities can be found and eliminated at an early stage. It is also possible to link that process to TARA, which is carried out as part of vehicle development. In principle, there is a high degree of similarity between a TARA and the approach shown here, as both processes aim to identify threats/vulnerabilities and attack paths. An applicability of the *Attacker Privilege* model explained in Figure 2 in the context of TARA has already been shown in [5]. However, our model-based security testing approach targets the testing process. We consider aspects, such as security mechanisms, as well as concrete exploits for potential vulnerabilities and detailed technological characteristics of vehicle systems. At the time of performing a TARA, such details are usually not yet available. One challenge of our approach is the transferability of attacks stored in our database to new vehicle systems or network architectures. In particular, if the network of a vehicle under test differs significantly from the network of an already attacked vehicle, there can be a risk that an attack path is not transferable. This problem can be circumvented by combining/permutating attack steps from different database attacks. Whether resulting attack paths actually reveal vulnerabilities in a vehicle, however, can only be determined by the tester. Furthermore, we want to highlight that it would make sense to carry out further testing activities. In general, our approach can be seen as a black-box test method. Even if we have detailed information about elements of the vehicle E/E architecture, our security model does not cover all aspects, such as software code. In case an attack path generated from our model reveals a vulnerability, we only know that there is a vulnerability. This does not mean that the root cause of that vulnerability is also known. Thus, additional grey-box or white-box-based test methods should be applied in this case to find the root cause. As a final aspect, we

 ISBN: 978-1-61208-996-6

discuss how our approach can be evaluated. In [5], we were already able to show that critical attack paths can be found by applying the *Attacker Privilege* model to vehicle architectures. In addition, we were able to determine in initial investigations that security models based on E/E architectures of attacked vehicles from our database (for example, from publications, such as Miller and Valasek [31]) contain new attack paths, which were also exploited in reality. These investigations should be extended to a detailed case study in future work.

VII. Conclusion

In this paper, an approach to enable automation of the security testing process was shown. In particular, we presented a formal security model, which can be analyzed for possible attack paths based on existing attacks from our attack database. We further demonstrated how paths for security testing can be derived. In addition, a proposal was presented on how knowledge and experience of penetration testers can be captured and reused to derive test paths. The approach is designed to deal with an increasing complexity of modern vehicles by automating sub-processes of the security testing process, in particular test planning and test case generation (attack paths). This enables early system analysis (e.g., as model-in-the-loop tests) and early testing. Further, estimations of security test effort can be made. For a practical implementation of the presented method, future work is to develop a software tool. This enables the creation of a security model, an analysis of that model, and the derivation of security testing paths. The tool can also be used to support penetration testers, as it provides knowledge about attacks or knowledge of testers in a comprehensive way. In addition, our approach should be evaluated in the context of a case study. Initial investigations have shown that existing attack paths from our database can also be found in other E/E architectures. A larger case study should therefore be carried out to examine this in detail.

Acknowledgment

This work was developed in the project SecForCARs-SAVE (reference number: 16KIS0796) which is funded by the German Ministry of Education and Research (BMBF).

References

[1] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, 2019.

[2] ISO/SAE 21434:2021, "Road vehicles — cybersecurity engineering," 2021.

[3] UNECE, "Un regulation no. 155 - uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system: E/ece/trans/505/rev.3/add. 154," 03/2021. [Online]. Available: https://unece.org/sites/default/files/2021-03/R155e.pdf (Accessed 2022.07.12).

[4] S. Marksteiner and Z. Ma, "Approaching the automation of cyber security testing of connected vehicles," in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–3.

[5] J. Dürrwang, F. Sommer, and R. Kriesten, "Automationin automotive security by using attacker privileges," in *Proceedings of the 19th escar Europe 2021*, pp. 137–152.

[6] F. Sommer, R. Kriesten, and F. Kargl, "Model-based security testing of vehicle networks," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 685–691.

[7] F. Sommer and J. Dürrwang, "Ieem-hska/aad: Automotive attack database (aad)," 2019. [Online]. Available: https://github.com/IEEM-HsKA/AAD (Accessed 2022.07.12).

[8] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-based security testing: a taxonomy and systematic classification," *Software Testing, Verification and Reliability*, vol. 26, no. 2, pp. 119–148, 2016.

[9] L. Wang, E. Wong, and D. Xu, "A threat model driven approach for security testing," in *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, 2007, p. 10.

[10] C. Nickerson et al., "The penetration testing execution standard," 2014. [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page (Accessed 2022.06.27).

[11] M. Meucci et al., "Owasp testing guide, v3," 2008. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v3.pdf (Accessed 24.08.2022).

[12] M. Ring, "Systematische security-tests von kraftfahrzeugen (systematic security tests of vehicles)," Dissertation, Universität Ulm, Ulm, 2019.

[13] S. Bayer, K. Hirata, and D. K. Oka, "Towards a systematic pentesting framework for in-vehicular can," *14th ESCAR Europe*, 2016.

[14] ISO 11898-1:2015, "Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling," 1993.

[15] ISO 14229:2006, "Road vehicles — unified diagnostic services (uds) — specification and requirements," 2006.

[16] C. Smith, *The Car Hacker's Handbook: A Guide for the Penetration Tester*. San Francisco: No Starch Press, 2016.

[17] J. Dürrwang, J. Braun, M. Rumez, R. Kriesten, and A. Pretschner, "Enhancement of automotive penetration testing with threat analyses results," *SAE International Journal of Transportation Cybersecurity and Privacy*, vol. 1, no. 11-01-02-0005, pp. 91–112, 2018.

[18] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, pp. 297–312, 2012.

[19] M. Cheah, S. A. Shaikh, O. Haas, and A. Ruddle, "Towards a systematic security evaluation of the automotive bluetooth interface," *Vehicular Communications*, vol. 9, pp. 8–18, 2017.

[20] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.

[21] Bluetooth Special Interest Group, "Bluetooth core specification v5.0," 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification (Accessed 2022.07.12).

[22] P. S. Oruganti, M. Appel, and Q. Ahmed, "Hardware-in-loop based automotive embedded systems cybersecurity evaluation testbed," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 2019, pp. 41–44.

[23] M. Appel, P. S. Oruganti, Q. Ahmed, J. Wilkerson, and R. Sekar, "A safety and security testbed for assured autonomy in vehicles," *SAE International*, p. 8, April 14, 2020.

[24] T. Volkersdorfer and H.-J. Hof, "A concept of an attack model for a model-based security testing framework: Introducing a holistic perspective of cyberattacks in software engineering," in *SECURWARE 2020 : The Fourteenth International Conference on Emerging Security Information, Systems and Technologies*, pp. 96–101.

[25] V. S. Alagar and K. Periyasamy, *Specification of software systems*, 2nd ed. Springer Science & Business Media, 2011.

[26] The MITRE Corporation, "Common weakness enumeration (cwe)," 2022. [Online]. Available: https://cwe.mitre.org/ (Accessed 2022.07.12).

[27] H. Booth, D. Rike, and G. Witte, "The national vulnerability database (nvd): Overview," Gaithersburg, 2013. [Online]. Available: https://www.nist.gov/publications/national-vulnerability-database-nvd-overview (Accessed 2022.07.12).

[28] L. Kohnfelder and P. Garg, "The stride threat model," 2009. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20) (Accessed 2022.07.12).

[29] The MITRE Corporation, "Common attack pattern enumeration and classification (capec)," 2022. [Online]. Available: https://capec.mitre.org/index.html (Accessed 2022.07.12).

[30] C. H. Liu et al., "Capture-replay testing for android applications," in *2014 International Symposium on Computer, Consumer and Control*, 2014, pp. 1129–1132.

[31] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, pp. 260–264, 2013.

 ISBN: 978-1-61208-996-6